APML - Convolutional Nets

יובל יעקבי, 302247077

2 בדצמבר 2018

חלק תאורטי

שאלה 1.1 - ReLU עם פרמטר

נגדיר: $f_i(o,t) = max(t, o_i)$, האם זה מגדיל את מחלקת ההיפוטזות שלנו?
לא!

הסיבה שלא כי זוהי בסה"כ פעולה לינארית שכל אחד מהמשקולות בגרף שלנו יודעת לעשות...

הוכחה קצת יותר פורמלית, נניח בשלילה שזה כן מגדיר מחלקה עשירה יותר, ויהיה A ארכיטקטורה כזו.

לכל שכבה שמשתמשת בReLU נוסיף נוירון אחד לפני האקטיבציה ואחד אחרי האקטיבציה, כאשר הנוריון שלפני מחסיר t והניורן שאחרי מוסיף t, ונשים לב שהארכיטקטורה הזו זהה לארכיטקטורה של A בלי להשתמש בפונקציה הנוספת...

שאלה 1.2 - נגזרת Sigmoid

נגדיר: $\sigma(x) = \frac{1}{1+e^{-x}}$
נתבונן בנגזרת:

$$\begin{aligned}
(\sigma(x))' &= \frac{\partial}{\partial x}\left(\frac{1}{1+e^{-x}}\right) = \\
&= \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1+e^{-x}-1}{(1+e^{-x})^2} \\
&= \frac{1}{(1+e^{-x})} - \frac{1}{(1+e^{-x})^2} \\
&= \sigma(x) - \sigma(x)^2 = \sigma(x)(1-\sigma(x))
\end{aligned}$$

כלומר אם במעבר הראשון נשמור את $\sigma(x)$, בback propagation כל מה שנצטרך לעשות זה כפל חיסור והעלאה בריבוע...

שאלה 1.3 - איתחול של רשת

עבור כל אחת מהמשקולות (u, w_1, w_2), הגראדינט מחושב כך:

$$\frac{\partial l}{\partial u} = \frac{\partial l}{\partial p} \cdot \frac{\partial p}{\partial u} = 2(p-y) \cdot m^T = 2(p-y) \cdot 0 = 0$$

כיוון שמאתחלים את כל המשקולות ל0, ויש פעולות כפל בין הניורונים נגיע למצב שתמיד $m = 0$, ולכן הגראדינט לא ישתנה בכלל ונשאר במקום.
בצורה דומה עבור w_1, w_2:

$$\frac{\partial l}{\partial w_j} = \frac{\partial l}{\partial p}\frac{\partial p}{\partial m_{:,j}}\frac{\partial m_{:,j}}{\partial o_{:,j}}\frac{\partial o_{:,j}}{\partial w_{:,j}} = 0$$

כיוון שגם כן יש לנו די הרבה משוקולות שפשוט ישארו 0 (למשל כל המטריצה של w).
כאמור הבעיה היא כמובן שהגראדינט ישאר אפס כל הזמן, ומכיוון שאנחנו מעדכנים את המשקולות בעזרת הגראדינט המשקולות לא יתעדכנו ונישאר במקום לאורך כל האימון.

חלק פרקטי

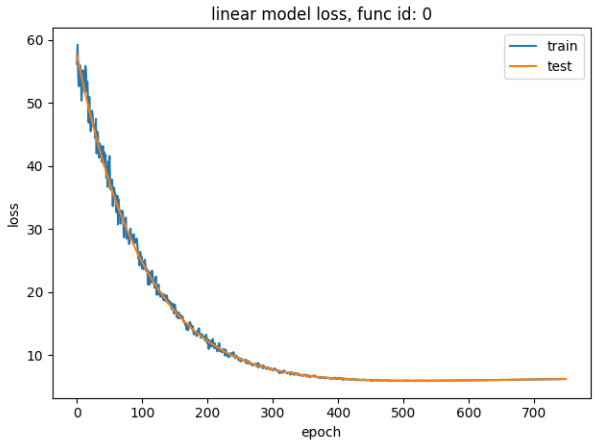
ארכיטקטורת ToyCovNet

לפני שניגש לגרפים המוכחים את הטענה נטען:

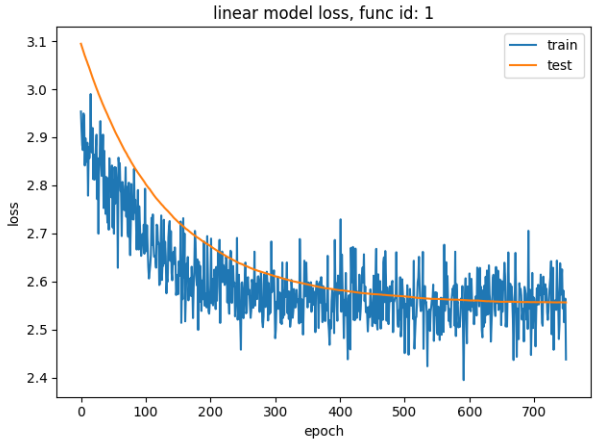
- הרשת הלינארית
  - תוכל ללמוד רק את הפונקציה הראשונה - $y_1(x) = \sum_i (Wx)_i$, כיוון שזוהי הפונקציה היחידה שלינארית, כלומר הנוירונים השונים ברשת יוכלו ללמוד את המטריצה W (נצטרך מספר נוריונים דומה למספר הכניסות במטריצה), וכך הרשת תוכל ללמוד את הפונקציה הזו
  - שתי הפונקציות האחרות אינן לינאריות ולכן הרשת לא תוכל ללמוד אותן
    * מקסימום - יכול כל פעם להיות במקום אחר, אין כאן בכלל משקולות לא משהו שרצף פעולות חיבור/כפל עם משקולות יצליח ללמוד
    * אם היו שכבות נוספות לרשת הלינארית היינו מצליחים ללמוד את זה, בשכבה אחת אי אפשר לללמוד את זה כיוון שיש פה כמה משקולות שתלויים נצטרך:
      · שכבה שתלמד לעשות סכום (כמו הפונקציה הראשונה שלמדנו)
      · שכבה שתעלה בחזקת e ותעשה סכום (אולי 2 שכבות)
      · שכבה שתוציא את הlog
- רשת קונבולוציה - הרבה יותר קשה לחזות מה הקונבולוציה תצליח לעשות, מכיוון שגם יש לה כמה שכבות, וגם פעולת הקונבולוציה היא עשירה יותר כנראה שנצליח ללמוד את כל הפונקציות...

גרפים

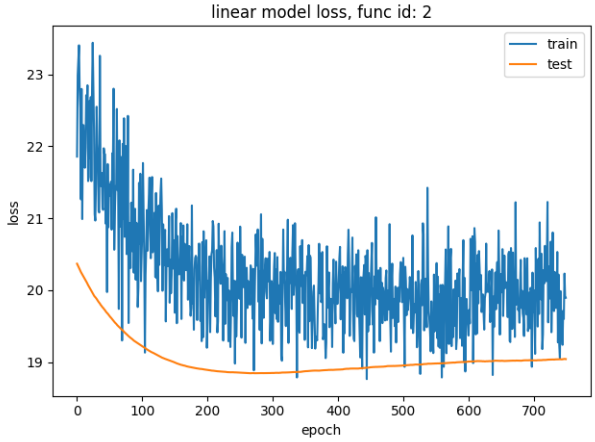
- רשת לינארית:



.1

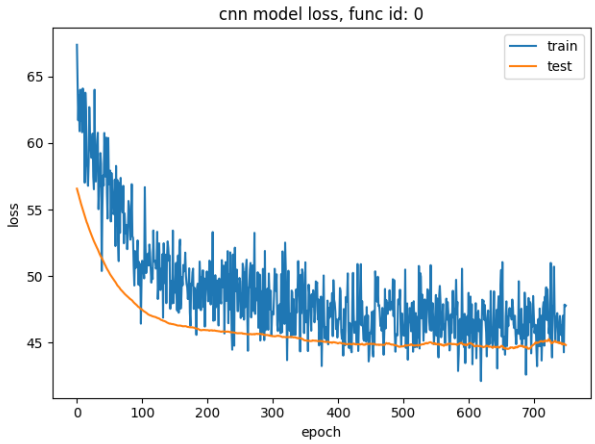


.2

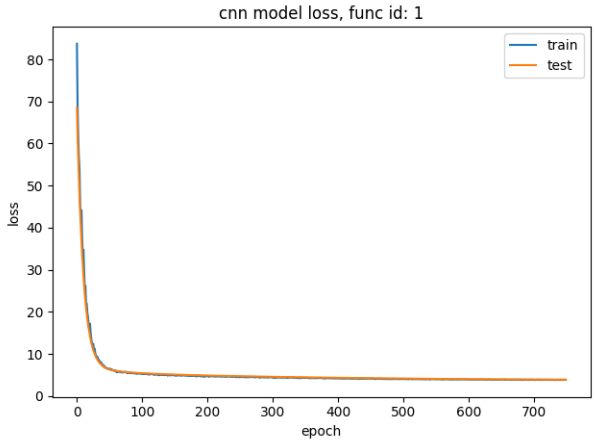


.3

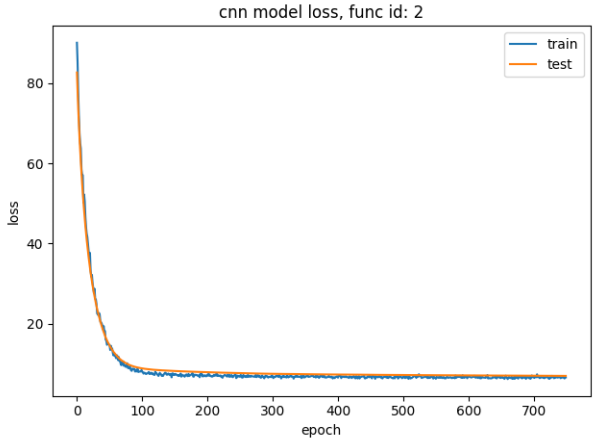
- רשת קונבולוציה:



.1



.2



.3

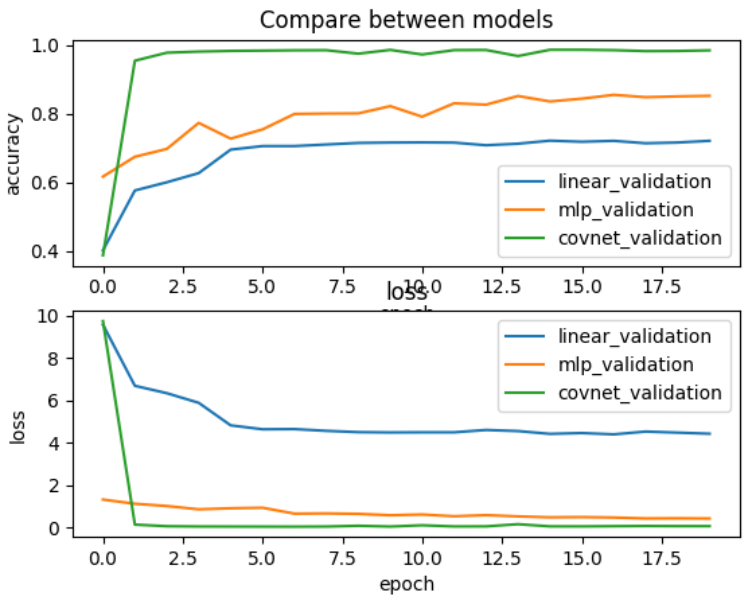
כפי שצפינו - הרשת הלינארית אכן למדה רק את הפונקציה הראשונה

בנוגע לקונבולוציה - אנחנו רואים שבפונקציה הראשונה היא הגיעה לרוויה מסוימת (כלומר לא מצליחה ללמוד עוד), אבל הloss עדיין יחסית גבוהה, יכול להיות שהרשת הזו "מסובכת" מידי לבעיה. בשאר הפונקציות אנחנו רואים שאכן מגיעים לloss יחסית נמוך.

גם כאשר מורידים את קצב הלמידה רשת הקונבולוציה לא מצליחה ללמוד את הפונקציה הראשונה, שזו תוצאה יחסית מפתיעה.

למידה על MNIST

הפעם נתחיל מגרפים שמשווים בין כל המודלים מבחינת הדיוק (Accuracy) וה loss



נסביר מה אנחנו רואים כאן:
הגרף העליון מתאר את הדיוק (אחוז התמונות שחזינו נכון), הגרף התחתון מתאר את פונקציית המחיר כאשר השתמשנו ב:categorical cross entropy
בציר x אנחנו מראים את מספר הepoch כלומר כל נקודה מראה על עדכון המשקולות מהגרפים אנחנו רואים:

- רשת הקונבולוציה "פתרה" את הבעיה יחסית מהר, כלומר זו רשת מספיק עשירה לקבל ייצוג מתאים של התמונות ולסווג אותן נכון.
- הרשת הלינארית לאחר כמה עידכוני משקולות מגיעה יחסית לרוויה (כלומר אנחנו לא מצפים שהיא תשתפר משמעותית גם אם נמשיך את האימון לעוד הרבה זמן)
- הרשת עם מספר השכבות הלינאריות לא לגמרי מתייצבת עדיין אבל ברור לגמרי שהיא טובה יותר מהרשת הלינארית ושהיא פחות טובה מרשת הקונבולוציה (שמגיעה לדיוק של 99 אחוז)

בחירות כלליות למודלים:

- השתמשתי באותה פונקציית מחיר לכל הרשתות ע"מ לנסות ולבצע את ההשוואה בצורה הטובה ביותר
- כנ"ל מבחינת "קצב הלמידה" (learning rate), השתמשתי בAdadelta שממצעת את גודל הצעד ע"פ הצעד הקודם, אני חושב שזו הדרך הכי טובה להשוות בין המודלים ככה גם אם מודל לומד "לאט" או "מהר" (כלומר צריך צעד למידה גדול או קטן), אזי הוא יקבל את זה דרך Adadelta ולא להשתמש באלגוריתם עם צעד קבוע
  - כאשר השתמשתי בSGD (קצב למידה קבוע) לרשת הקונבולוציה היה מאוד קשה ולא הצליחה ללמוד כלום, כנראה שהיא הלכה מסביב למינימום שהיא חיפשה...

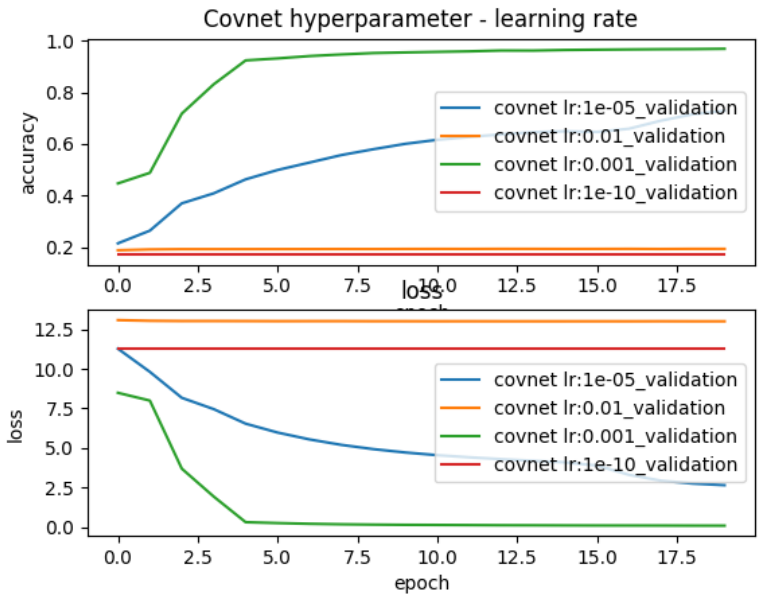
בחירות פר מודל:

- במודל הלינארי הבחירה היחידה היא האקטיביציה על השכבה האחרונה, בחרתי בSoftmax כנהוג לשכבה אחרונה כדי שסכום ההסתברויות יהיה 1
- במודל המולטי לינארי - כמצופה הוא טוב יותר מהמודל הלינארי (נניח בשלילה שהלינארי טוב יותר אזי המודל הזה ימשקל את כל המשקולות ב1 חוץ מהשכבה האחרונה שזהה למודל הלינארי)
  - מבחינת פונקציית אקטיביציה, בשלבים שבדרך בחרתי בReLU פונקציה לא לינארית שלא סוכמת לאחד (ובשכבה האחרונה Softmax כאמור)
  - מבחינת כמות השכבות - אין סיבה מיוחדת לבחירה, המחשבה הייתה להוריד לאט לאט את כמות המימדים של התמונה כך שבכל שכבה נלמד פ'ייצרים מדויקים יותר, בפועל מהניסויים שעשיתי לא נראה שזה היה משנה והרשת מגיעה לתוצאות זהות גם עם מימדים אחרים בכל שכבה
- קונבולוציה - שיקולים דומים לרשת המולטי לינארית, גם כאן ברוב הארכיטקטורות שניסיתי התוצאות היו דומות (ומצוינות כאמור)
  - השכבה האחרונה היא שכבת "שיטוח" Flatten כדי לרדת מתמונות דו מימדיות לוקטור תוצאות שיהיה חד מימדי

אופטימזציה על פרמטרים - קצב למידה Learning Rate

בדקנו על אותה רשת קונבולוציה שהשתמשנו בה כדי ללמוד את mnist בסעיף הקודם, השתמשנו הפעם באלגוריתם SGD והשתמשנו בקצבי למידה שונים.

קצת על המצופה - ניסיתי להשתמש בטווח יחסית רחב של תוצאות, מצד אחד אני מצפה שיהיו צעדים גדולים מידי שבהם לא נלמד כלום כי נלך הלוך חזור סביב המינימום, בצד שני צעדים קטנים מידי שלא יתקדמו לשום מקום ולא נצליח ללמוד באמת, ובאמצע את הקצב המוצלח עבור הבעיה הספציפית:



כפי שניתן לראות באמת יש לנו כאלה

- הגרפים הכחול והירוק, קצב למידה באמצע שהוא מספיק טוב על מנת שהרשת אכן תלמד את הבעיה, הירוק עושה את זה טוב יותר כי יש לו גודל צעד גדול יותר, כנראה שאם היינו עושים עוד צעדי למידה (עוד epochs) אז שניהם היו מגיעים לאותם ביצועים
- הגרפים האדום והכתום, הינם משני הצדדים, האדום עם גודל צעד קטן מאוד, הכתום עם גודל צעד גדול, שניהם אנחנו רואים שנשארים סטטים עם דיוק מאוד מאוד נמוך... כנראה שאם היינו עושים המון המון צעדים אז האדום היה מצליח ללמוד, לעומת זאת הכתום לא היה מצליח ללמוד...

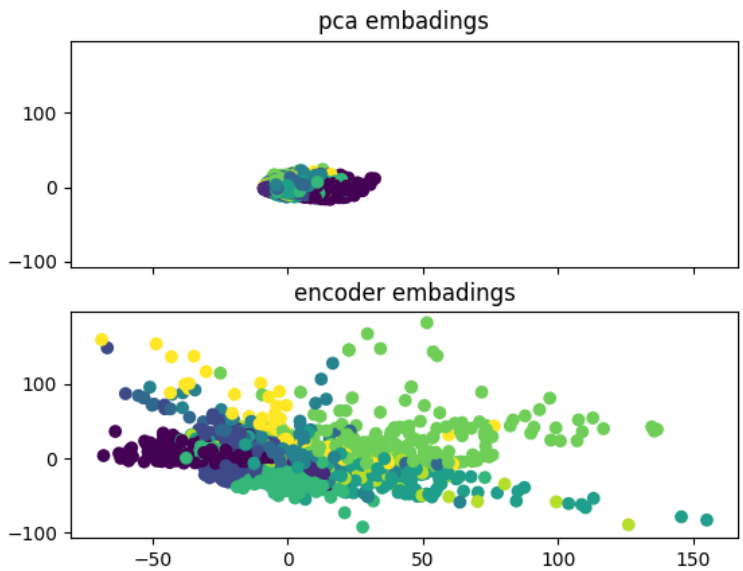
מקודד AutoEncoder

צורת מימוש:

- השתמשתי רק בשכבות לינאריות
- כאמור ניתן לחלק את הרשת לשניים, החלק שמקודד והחלק שמפענח, בניתי אותם בצורת מראה כיוון שזה מה שאנחנו מנסים לעשות
- מבחינת האקטיבציה, בחרתי שוב בReLU כדי שיהיה פונקציה לא לינארית בתהליך, אחרת התהליך הוא דומה לPCA
  - באמת אם בוחרים באקטיבציה לינארית התוצאות שוות לPCA גם מבחינת הגרף וגם מבחינת המחיר

התוצאות של הautoencoder טובות יותר מבחינת המחיר התוצאות שקיבלנו (מרחק ריבועי):

- על הדאטא של ה"אימון" -
  * PCA=0.826
  * autoencoder=0.722
- על דאטא שלא ראינו באימון (טסט):
  * PCA=0.764
  * autoencoder=0.665
- נתבונן בקידוד שעשינו (במימד 2),



- כפי שניתן לראות מהגרף הקידוד שעשינו באמצעות הרשת משמעותית טוב יותר, הוא מתפזר טוב יותר במרחב כמעט בקלאסטרים נפרדים